Memoria del Trabajo de Fin de Grado

Diseño y Evaluación de Algoritmos de Análisis de Sentimiento y Clasificación de Texto a partir de Tweets

Alvaro Martínez López-Zuazo

9/27/2015

Índice de contenidos:

1. Motivation:

Currently, it is said that Internet has evolved into "Web 2.0", a network whose major contributors when it comes to providing information and contents are the netizens. This is due to the boom in services that keep information on the network, either to exchange it with other users (like social networks) or for a more personalized experience.

So Internet is a rich source of user-generated information, and that well-analyzed information can be very useful to companies besides netizens. Therefore, research areas such as opinions mining or Sentiment Analysis, which are going to be described next, they have experienced a parallel rise of those services.

To understand the opinion mining concept is first necessary to know the natural language processing (NLP) concept. The NLP is the design of algorithms that allow establishing a computationally efficient man-machine communication through a natural language. I.e, try the machine is able to learn and interpret several terms included in a sentence. It´s not easy work, because interpretations of natural language in many cases can be ambiguous for the machine, in example: the same word can have several meanings, one sentence can be ironic, reception data is not always perfect (words misspelled, ungrammatical expressions). So when it comes to natural language processing, 5 phases are necessary:

1. Morphological analysis: words are detected individually, as well as components that are not part of them, such as punctuation marks.

2. Parsing: structures of words that relate to each other are created, and they will be rejected if the natural language words that make up the structure cannot be combined as they are doing.

3. Semantic analysis: We check if not previously rejected structures have a meaning. Those that don't have any meaning are rejected too.

4. Speech Integration: Analyze whether the meaning of individual words or phrases influences others or depends on some precedent.

5. Pragmatic Analysis: Message interpretation.

Knowing this, then I'm going to proceed explaining what is opinion mining and why it is a piece of knowledge with great prospects for researchers and companies and users.

When we talk about opinión mining we refer to extracting by natural lanuage processing, subjective information about user generated content. For example, determining whether a review is negative or positive, which is very useful when performing a markt analyss, since it's a way to control the affinity or dissatisfaction degree about a product. Furthermore, classifying opinions can be helpful when interest focusing on those opinions with a certain polarity (may interest only negative or positive opinions).

The Sentiment Analysis is a further step in this field, also in complexity, since it consists in trying to determine the characteristics of a product and extract the polarity of each. This would allow a post hoc analysis of its strengths and areas for improvement points, a consumers perspective-based analysis, since they ultimately are who decide if a product is sold or not.

In addition to a marketing.oriented use, with seniment anaylisis we can use information through opinions on other topics of interest like policy (polling estimation), economy (analysis of critical situations), hobbies, etc.

2. Análisis de sentimiento a partir de las opiniones de Tweets

Dentro de las redes sociales, Twitter se ha convertido en la principal plataforma a la hora de opinar sobre cualquier tópico, por lo que los investigadores en el ámbito de minería de opiniones y análisis de sentimiento están dedicando grandes esfuerzos en el análisis de los tweets.

Un ejemplo de esto es el hecho de que en los últimos años se han organizado bastantes más tareas y concursos relacionados con el análisis de sentimiento, como los **Talleres TASS:**

Se trata de Talleres para la evaluación experimental del Análisis de Sentimientos y Seguimiento de opinión en redes sociales. En estos talleres se proponen una serie de tareas relacionadas con el análisis de sentimiento para fomentar el diseño de algoritmos de clasificación relacionados con esta cuestión, comparando posteriormente las diferentes implementaciones de los sistemas desarrolladas por los participantes.

Esto tiene fines científicos, se trata en última instancia de obtener en cada edición el mejor análisis del sentir social, todo ello a partir de las opiniones expresadas en Twitter. Las tareas propuestas son las siguientes:

1. Clasificación de los tweets en tópicos: Consiste en asignar correctamente uno o varios tópicos a los tweets, siendo el corpus el mismo de la tarea 4. En la siguiente página se muestran las categorías posibles, así como la distribución de los tuits por tópico:

TÓPICO	%
Cine	2.56
Deportes	1.18
Economía	9.84
Entretenimiento	17.53
Fútbol	2.63
Literatura	1.08
Música	5.91
Otros	24.41
Política	32.59
Tecnología	2.27
TOTAL	100

2. Detección de Aspectos: Se trata de detectar aspectos de diferentes temáticas de interés en los tweets. Un aspecto es un concepto, un nombre, que puede escribirse de diferentes formas pero hace referencia a la misma entidad, con una polaridad asociada. Por ejemplo, supongamos que queremos detectar todas las referencias al futbolista Leo Messi, entonces una detección aceptable del aspecto sería aquella que englobara, entre otras, las siguientes posibilidades:

   Messi, messi, MESSI, mesi, Leo, Lionel, Leoo, LEO, #Messi, @Messi, @TeamMessi, #Messisinvergüenza, @MessiSiempreElMejor, @Leo_Messi_Barca

   Para esta tarea, en 2014 se proporcionó el corpus Social TV, que se compone de un conjunto de aproximadamente 3000 tweets recolectado durante la final de la Copa del Rey 2014. La tarea consistió en detectar en los tweets las referencias a los distintos jugadores, entrenadores, equipos, medios de comunicación, etc. que en ella participaron.

3. Análisis de polaridad de Aspectos de Twitter: Depende necesariamente de la tarea 2, ya que consiste en añadir polaridad a los aspectos detectados en dicha tarea.

4. Análisis de sentimientos en tweets: Consiste en determinar la polaridad de los tweets, desde 2 enfoques diferentes:

   1º) Distinguiendo entre 6 etiquetas de polaridad (N, N+, NEU, NONE, P, P+).

   2º) Distinguiendo entre 4 etiquetas de polaridad (N, NEU, NONE, P).

   Para esta tarea se utiliza un corpus de aproximadamente 70000 tweets en español, escritos por personalidades de distintos entornos (políticos, artistas, periodistas, etc.). Estos tweets están contenidos en varios ficheros como líneas, en las que se incluyen características de los tweets como el id del tweet, el usuario creador del mismo, el texto del tweet y una etiqueta inicial que indica la polaridad del tuit (P+, P, NEU, N, N+).

   Esta tarea es en la que nos vamos a centrar en el desarrollo propuesto en este proyecto.

2.1 Estado del arte:

Actualmente, existen diversas herramientas de procesado del lenguaje. Un ejemplo de ello son las COES, desarrolladas por el Departamento de Arquitectura y Tecnología de Sistemas Informáticos (DATSI) de la UPM junto al Departamento de Informática de la UC3M. Estas herramientas incluyen un buscador de palabras en un contexto, un diccionario de sinónimos, entre otros. Para más información, se puede acceder a estas herramientas en el siguiente enlace:

http:/www.datsi.fi.upm.es/~coes/

Para el caso que nos ocupa, debido a la especial naturaleza de Twitter (longitud limitada de caracteres, uso de un lenguaje no estándar, sintaxis especial: RT, emoticonos, @user, #tag… y otros fenómenos, las herramientas de procesado del lenguaje habituales pueden no resultar de gran utilidad. Pero este es un problema que podemos resolver mediante una normalización del texto como preprocesado del análisis.

Una vez se han normalizado, si que se pueden utilizar las herramientas tradicionales de NLP para analizar los tweets y extraer características que sean indicativas de la polaridad de los mismos. En cuanto a las características (features) que se pueden extraer, hay bastantes enfoques distintos:

- TF-IDF: Se explicará en profundidad en la parte del flujo de trabajo del proceso de clasificación.

- Lemas: Conjunto de palabras que representan el mismo concepto, a las que puede asignarse por tanto un patrón común y que tendrán la misma polaridad.

- POS (part of speech) tags: El etiquetado gramatical consiste en inferir el tipo gramatical de cada término, es decir, etiquetarlo como sustantivo, verbo, adjetivo, adverbio, pronombre, conjunción, preposición, artículo o interjección. De entre estas categorías cabe destacar que los sustantivos, artículos, pronombres y adjetivos son lemas, puesto que varían en género y número pero representan la misma idea. Mediante este enfoque, por ejemplo, se agruparían en un mismo diccionario aquellos términos que pertenecieran a la misma clase gramatical, cada uno con su correspondiente etiqueta de polaridad. La aproximación de aprendizaje automático más utilizada para la extracción de este tipo de características son los Modelos de Markov ocultos, también conocidos como n-gramas.
  Dicha técnica se basa en construir probabilidades de que en una secuencia de n palabras se den unas categorías gramaticales determinadas. Por ejemplo, supongamos que tenemos el siguiente bigrama (2 palabras): “el análisis”. Si detectamos que la primera palabra es un artículo, la siguiente palabra es mucho más probable que sea un sustantivo al resto de categorías.
  Esta información es útil a la hora de extraer otras medidas como el TF-IDF de los términos, puesto que se puede calcular el TF-IDF de un n-grama completo, que puede contener por ejemplo, una “expresión hecha”.
  Este enfoque ha sido aprovechado por numerosos investigadores, como Barbosa y Feng (2010) y dadas sus características se ha combinado con el cálculo del TF-IDF en otros diseños (Saralegi y San Vicente).

- Otros buenos indicadores de polaridad son los emoticonos, (O' Connor), siempre combinados con otra medida.

- Otro enfoque podría ser el de análisis de opiniones a nivel de discurso, es decir, teniendo en cuenta que un discurso puede contener varias opiniones. En un mismo discurso pueden existir, por tanto opiniones negativas y positivas acerca de ciertos 'targets' o entidades, con lo que el estudio que se realiza es el de las relaciones entre dichos targets y el sentimiento que expresan en cada contexto (Somasundaran, Wiebe, Ruppenhofer, 2009).

- Otra interesante posibilidad es tratar de inferir la polaridad de los tweets teniendo en cuenta los followers del autor. Para este enfoque, se asume que la gente puede influir en las opiniones de otra gente cuando comparten gustos acerca de temas concretos (Speriosu, 2011).

Además de todos estos indicadores de polaridad, se pueden tomar distintas estrategias:

- Aproximación semisupervisada combinando un diccionario de polaridad con propagación de etiquetas (Sindhwani and Melville, 2008). La propagación de etiquetas es un algoritmo similar al KNN cuya idea es asignar a las muestras sin etiquetar, la misma clase que a las muestras etiquetadas a las que sean más cercanas:

- Otra posible estrategia pasa por combinar diccionarios de polaridad con aprendizaje máquina para etiquetar el sentimiento. (Kouloumpis, Wilson y Moore, 2011; Saralegi y San Vicente, 2012; Mohammad, kiritchenko y Zhu, 2013). Esta aproximación es la más utilizada, y a lo largo de las ediciones del TASS se ha probado que es bastante exitosa, por lo que es la que se ha utilizado en este proyecto. Para realizar la clasificación mediante este enfoque, se pueden seguir varios esquemas, por ejemplo hacer que la máquina aprenda un clasificador binario para cada clase, de manera que sea capaz de distinguir entre esa clase y todas las demás (Hurtado, Pla, 2014). A esta estrategia se la conoce como clasificación uno contra todos (one-vs-all).

2.2 Estrategia seleccionada:

Se ha utilizado un esquema de clasificación basado en la medida del TFIDF de palabras, lemas y n-gramas preprocesados, partiendo de la creación de diccionarios de TFIDF a partir de las frecuencias de los términos dentro de conjuntos de tweets seleccionables en cuanto a tamaño. El procesado se realizará tweet a tweet para disminuir la carga computacional También se han tenido en cuenta otros indicadores como por ejemplo los emoticonos. Se ha entrenado el clasificador utilizando la técnica de validación cruzada con 10 folds para ajustar los parámetros y se ha estimado un intervalo de confianza para un nivel de confianza de la medida de éxito de un 95%. E

En los siguientes puntos se procederá a describir esta solución en profundidad, así como los resultados obtenidos.

3. Diseño de la solución técnica

3.1 Diseño del hilo principal

Se trata de un algoritmo que extrae las características de los tweets para posteriormente poder procesarlos y estimar la polaridad de los mismos utilizando sus medidas de TF-IDF.

El corpus disponible proporcionado por el TASS consta de:

- Un conjunto de 7219 tweets de entrenamiento. En dicho conjunto, la distribución de tweets según su polaridad es la siguiente:

POLARIDAD	Nº TWEETS	%
N	1335	18.49
N+	847	11.73
NEU	670	9.28
NONE	1483	20.54
P	1232	17.07
P+	1652	22.88
TOTAL	**7219**	**100**

- Un conjunto de 60798 tweets de test.

Es decir, el nº de tweets con los que se entrena el clasificador es muy inferior al nº de tweets con los que se realiza el experimento y podrían ser insuficientes. Una posible solución a esto es utilizar la técnica de validación cruzada, que consiste en lo siguiente:

1. Dividir los datos de prueba en N subconjuntos, de ahí que esta técnica se denomine también "N-fold cross validation".

2. Una vez hecha la división se utiliza uno de los subconjuntos como datos de test (estos datos conforman el subconjunto de validación), y con el resto se entrena al clasificador. Este proceso se repite N veces, es decir, se utiliza cada vez un subconjunto distinto como datos de test y el resto de subconjuntos como datos de entrenamiento, hasta haber utilizado todos los subconjuntos para test. A modo de ejemplo, la siguiente tabla muestra cómo se realizaría la división en 5 subconjuntos de un conjunto de datos:

Iteración	Conjunto1	Conjunto2	Conjunto3	Conjunto4	Conjunto5
1	Validación	Train	Train	Train	Train
2	Train	Validación	Train	Train	Train
3	Train	Train	Validación	Train	Train
4	Train	Train	Train	Validación	Train
5	Train	Train	Train	Train	Validación

3. Finalizadas las iteraciones correspondientes, se realiza la media aritmética del resultado de interés (que para el caso que nos ocupa será el 'Porcentaje de éxito de la estimación'), y se obtendrá un único resultado promedio, que estará comprendido en un cierto intervalo de confianza, concepto que se tratará más adelante.

El formato de los tweets en los ficheros de train y test es el siguiente: Cada línea contiene las características (se suelen nombrar como features) del tweet al que hace referencia separadas por el carácter **";"**, por ejemplo:

```
84;142928818708029440;aarbeloa17;1787997;0,0,0,0,0,0,0,0,1,1;P;AGREEMENT;Gracias por la hospitalidad!!! ;-) RT
@IsabelaTortlini Bienvenido DON @aarbeloa17 a mi humilde morada ;)
85;142933511345668097;cucagamarra;2093;0,0,0,0,0,0,0,0,0,1;P;AGREEMENT;En el Espolón, conmemorando el Día de
la Discapacidad http://t.co/52olz4vl
86;142935206737887232;mariviromero;13260;0,0,0,0,0,0,0,0,0,1;NONE;AGREEMENT;Así es el belén del Ayuntamiento
http://t.co/S6Ocgz6i
87;142935376787554304;mariviromero;13260;0,0,0,0,0,0,0,0,0,1;N+;AGREEMENT;Los vándalos le arrancan de nuevo el
lápiz a Picasso http://t.co/LfyNuArl
88;142936416949780481;mariviromero;13260;1,1,0,0,0,0,0,0,0,0;P;AGREEMENT;Esto es apoyar de verdad: La
@DiputacionMLG  reducirá impuestos a 96 pueblos http://t.co/VIGtaT1k
89;142937245148651521;cesc4official;4788777;0,0,0,0,0,0,0,0,1,0;N;AGREEMENT;Un poco d agua fria despues dl
entrenamiento esta mañana. Ahora a comer un poco d pasta y descanso total para esta noche q sera muy dificil!
90;142938216415240192;Buenafuente;1332368;0,0,0,1,0,0,0,0,0,0;P;AGREEMENT;Que corra el Dewars!!!"@DewarsSpain:
@jordievole @buenafuente @julia_otero @santi_millan gracias Jordi"
```

De manera que si nos centramos, por ejemplo, en el tweet 90, sus features son:

FEATURE	**SIGNIFICADO**
"90"	***Nº del tweet en la lista***
"142938216415…"	***ID del tweet***
"Buenafuente"	***ID del usuario***
"1332368"	***Nº followers del usuario***
"0,0,0,1,0,0,0,0,0,0"	***Tópico, tema del que trata el tweet***
"P"	***Polaridad del Tweet***
"AGREEMENT"	***N.A***
Que corra el Dewars!!! "@DewarsSpain: @jordievole @buenafuente @julia_otero @santi_millan gracias Jordi"	***Texto del tweet***

NOTA: Los ficheros se guardarán como: '…/intermediate/test/<tweet+nºtweet>.txt'

Por último se guardarán todas las líneas leídas (que procederán de los distintos ficheros) en un array.

El proceso principal se lanza desde línea de comandos, ejecutando un fichero denominado 'TASS2015-weka-testing.py' con diferentes opciones:

Tamaño de los diccionarios (opción --dicsize):

El algoritmo utiliza 3 tipos de diccionarios:

- Diccionarios de palabras: Ficheros.txt que contienen n términos (donde n es el tamaño del diccionario) del fichero de

entrenamiento, así como su frecuencia en el mismo. Pasarán a los diccionarios de tfidf sin cambios.

- Diccionarios de lemas: Los lemas son patrones que se aplican para varias palabras que comparten raíz, por ejemplo:

```
[34][anuncia][>][anunci]
[1][anunciada][>][anunci]
[1][anunciadas][>][anunci]
[7][anunciado][>][anunci]
[1][anuncian][>][anunci]
[6][anunciar][>][anunci]
[4][anunciara][>][anunci]
[1][anunciaremos][>][anunci]
[2][anuncie][>][anunci]
[14][anuncio][>][anunci]
[3][anuncios][>][anunci]
```

De manera que cuando se utilizan estos diccionarios (se debe activar la opción --use stems) si se detecta que hay términos en un tweet que tienen patrón se sustituyen por el mismo, y posteriormente en la medida del TFIDF se usarán diccionarios de TFIDF que contengan lemas.

NOTA: El número que aparece al lado de cada palabra es la frecuencia de la misma en el fichero de entrenamiento. Se utilizará para posteriormente calcular el TFIDF.

- Diccionarios de TFIDF: Ficheros.txt que contienen n términos (donde n es el tamaño del diccionario) de cada clase, junto a los scores de TFIDF obtenidos por cada uno de los términos. De modo que si tenemos 6 clases existirán 6 ficheros, cada uno compuesto por palabras etiquetadas para una cierta clase. Por ejemplo, para la clase 'N' el diccionario de 500 términos sigue el siguiente formato:

```
[0.00025676872775][bomba][bomba]
[0.000225300115637][despido][despido]
[0.000221245844197][asesinatos][asesinatos]
```

La opción --dicsize permite seleccionar un tamaño para los diccionarios que utiliza el proceso, por ejemplo:

"python TASS2015-weka-testing.py --root_filename PRUEBA2015 --baseline --dicsize 2000"

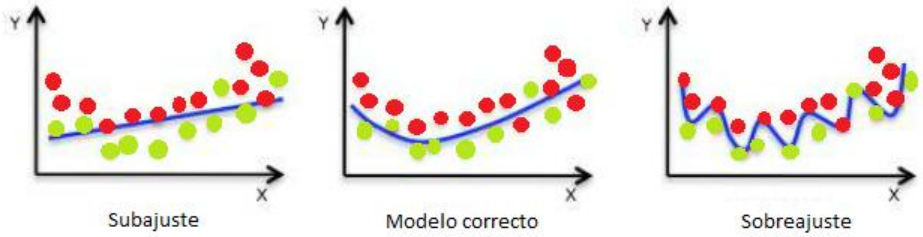
La anterior ejecución utilizará diccionarios de 2000 términos durante el proceso. Si no se selecciona ningún tamaño, el proceso tiene uno por defecto de 500.

El tamaño de los diccionarios es un factor importante en cuanto a los resultados del experimento, ya que escoger un tamaño pequeño (es decir, utilizar diccionarios de pocos términos) hace que el clasificador subajuste, y del mismo modo, utilizar un tamaño muy grande podría hacer que el clasificador sobreajustase. Ambos fenómenos influyen negativamente en los resultados del análisis:

- El subajuste se produce cuando no entrenamos al clasificador con un nº de muestras suficientes, por lo que cuando encuentra muestras que no haya 'visto' en el entrenamiento no es capaz de clasificarlas correctamente. En nuestro caso el subajuste se producirá cuando utilicemos un diccionario de pocos términos, por ejemplo, el de 500, ya que cuando los tweets contengan términos fuera de esos 500 el clasificador no será capaz de detectar su polaridad correctamente. Por ejemplo, en el caso de los diccionarios de TFIDF se da la situación de que a medida que crecen en tamaño los diccionarios, pueden contener términos que para un tamaño más pequeño pertenecieran a otra clase. Por ejemplo, el término 'estupendo' no está en ningún diccionario asociado a la clase P para un tamaño de 500 términos, pero si escogemos un tamaño mayor (p.ej: 3600) sí que existe en dicha clase, con lo cual el TF-IDF obtenido para la clase P en el mismo término sería distinto en ambos casos.

- El sobreajuste se produce cuando el clasificador se ajusta muy bien al conjunto de datos con que se ha entrenado, pero sin embargo cuando llegan nuevas muestras no decide bien, puesto que está demasiado ajustado a los datos iniciales. En nuestro caso, el sobreajuste se produce cuando utilizamos diccionarios demasiado grandes, porque contienen tantos términos que puede que haya términos con mayores scores en clases a las que realmente no pertenecen. Por ejemplo, puede haber un término en P+ que también esté en P con menor TFIDF y por tanto que el tweet sea detectado como P+ cuando su etiqueta original era P.

A continuación se muestran las representaciones gráficas más usuales de los fenómenos de sobreajuste y subajuste:



Donde las muestras rojas representan una clase y las verdes otra (por ejemplo, 'P' y 'P+'), y la línea azul es la frontera establecida por el clasificador.

Detección de emoticonos (opción --emoticons):

Activa el parseo de emoticonos, de manera que pasan a integrarse dentro de la estructura TWEET. Para ello:

1. Se dispone de una clase 'EMOTICONS' con una serie de arrays, conteniendo cada uno emoticonos asociados a distintos estados de ánimo:

EMOTICONOS	ARRAY
:-\) :\) :o\) :\] :3 :c\) :> =\] 8\) =\) :\} :\^\) :ɔ\)	smileys
:-D :D 8-D 8D x-D xD X-D XD =-D =D =-3 =3 B^D	laughs
:-\)\) :\'-\) :\'\)	very_happy
>:\[:-\(:\(:-c :c :-< :ɔC :< :-\[:\[:\{	sadness
:\'-\(:|'\(QQ	crying
:-|| :@ >:\('	angry
D:<' D:' D8 D; D= DX v\.v D-\':	horror_disgust
>:0 :-0 :0 \W ºoº ºOº :0 \W+ o_O o_0 o\.O \W+ 8-0	surprise_shock
:* :\^* \(\'\}\{\' \)	kiss
;-) ;) *-) *) ;-] ;] ;D ;^) :-,	wink
>:P :-P :P X-P x-p xp XP :-p :p =p :-P :b	cheeky_playful
>:\\ >:/ :-/ :-. :/ :\\ =/ =\\ :L =L :S >.<	skeptical_annoyed
:| :-|	straight_indecision
:\\$	embarrassed_blushing
:-X :X :-# :#	sealed_lips

2. Después de parsear los tweets y antes de dividirlos por palabras estos pasan por un bloque de preprocesado. Dentro de las funciones de este bloque, se incluye la de detección de emoticonos, que se produce a partir de la llamada a un método denominado 'ParseTweetEmoticons', cuya lógica se basa en lo siguiente:

En la clase 'EMOTICONS', los arrays de emoticonos se agrupan en 3 arrays más grandes según la polaridad que expresan:

- positive_emoticons= Incluye los emoticonos de los arrays: smileys, laughs, very_happy, kiss y wink.

- negative_emoticons= Incluye los emoticonos de los arrays: sadness, crying, angry, horror_disgust, surprise_shock, skeptical_annoyed, straight_indecision, embarrassed_blushing y sealed_lips.

- cheeky_emoticons= Incluye los emoticonos del array cheeky_playful

De manera que dependiendo del grupo en el que se encuentre el emoticono, se le asigna un patrón:

- positive_emoticons ------> **'ffmemopos'**

- negative_emoticons ------> **'ffmemoneg'**

- cheeky_emoticons ------> **'ffmemocheeky'**

'ParseTweetEmoticons' contiene 3 llamadas (una por cada patrón de emoticonos) al método ProcessSpecificPatern, que es el que finalmente se encarga de detectar si hay emoticonos pertenecientes a los grupos en los tweets y, en caso de que los haya, sustituirlos por su patrón correspondiente en el texto. Para ello, el proceso va buscando los emoticonos de cada grupo uno a uno.

A continuación un ejemplo del procesado de emoticonos:

1. En primer lugar, vamos a introducir un tweet por teclado que contenga un emoticono:

```
[CreateSingleTweet] [CreateTweet] [Tweet 1 of 1]
        Write the tweet: Ya es viernes!! :)

        1-politics
        2-economy
        3-music
        4-entertainment
        5-literature
        6-sports
        7-technology
        8-movies
        9-football
        10-others
        Select 1 topic: 10

        P+
        P
        NEU
        N
        N+
        NONE
        Write the Polarity: P+
```

2. El proceso empieza a buscar uno a uno los emoticonos en el tweet, y cuando encuentra uno lo sustituye por su patrón:

```
[NEW][Ya es viernes!! :)]
[AFTER 0 'ffmemopos' RESULTS]
[Press key to continue...]

[NEW][Ya es viernes!! ffmemopos ]
[AFTER 1 'ffmemopos' RESULTS]
```

La imagen representa cómo en el primer intento, no se ha encontrado en el tweet el emoticono que se estaba buscando, pero en el siguiente intento se encontró ':)', por lo que se ha sustituido en el texto del tweet por su patrón (que es el que corresponde a emoticonos positivos). Además, el proceso informa por pantalla de que se ha encontrado el emoticono:

```
[1 ffmemopos FOUND!!!]
[1][:)]
```

3. Además, se genera un archivo que contiene todos los emoticonos positivos parseados (en este caso, sólo contiene ':)'

Este mismo preprocesado se puede realizar con:

- **Hashtags -->** Se les asigna el patrón **hashtagffm:**

```
[POSSIBLE HASHTAG AFTER REMOVING QUOTES AND SPACES]
[#estupendo]

Twit labels:
[hashtagffm]
[TWIT][1][UPDATED!!!]
[NEW][Hola me parece hashtagffm]
```

De manera que cuando se detecta este patrón durante el procesado, se descarta el carácter # a la hora de mapear el término.

Al igual que en el caso de otros patrones que se preprocesan (como emoticonos, números o URLS), se genera un archivo con todos los hashtags detectados.

- **Números -->** Se activa mediante la opción --num_parsing. Se les asigna el patrón **ffmdigits**, por ejemplo, para el tweet introducido por teclado "No te lo digo 2 veces, ordena tu cuarto!" se muestra por pantalla:

```
[TWIT][1]
[ORI][No te lo voy a decir 2 veces, ordena tu cuarto!]
[CUR][No te lo voy a decir 2 veces, ordena tu cuarto!]

[NEW][No te lo voy a decir ffmdigits veces, ordena tu cuarto!]
[AFTER 1 'ffmdigits' RESULTS]
```

- **URLs -->** Se les asigna el patrón **urlffm.**

Procesado individual de tweets (opción --single_tweet):

Activa el modo de procesado individual de tweets. Este modo de funcionamiento se ha pensado de cara a reducir la carga computacional que supone el procesado de un nº muy elevado de tweets en un mismo equipo. Individualizando el procesado, se podría ejecutar el proceso de manera mucho más rápida ya que se podría 'clusterizar', es decir, enviar cada tweet a un componente de clúster

distinto para realizar sus procesados en paralelo en lugar de sólo una vez.

Al seleccionar esta opción, se permite la introducción de tweets por teclado. Para ello:

- En primer lugar, se selecciona el nº de tweets que se van a introducir:

```
Select written tweets number: 2
```

- Una vez seleccionado, se realizarán tantas veces como tweets se vayan a introducir los siguientes pasos:

1. Introducir el tweet: Por ejemplo, 'hola me parece estupendo'.
   NOTA: Aunque el ejemplo sea sencillo, el proceso también permite la introducción de hashtags, RTs, emoticonos y citas.

```
[CreateSingleTweet] [CreateTweet] [Tweet 1 of 2]
        Write the tweet: hola me parece estupendo
```

2. Introducir el tópico del que trata el tweet: Por ejemplo, '10-others':

```
1-politics
2-economy
3-music
4-entertainment
5-literature
6-sports
7-technology
8-movies
9-football
10-others
Select 1 topic: 10
```

3. Introducir la polaridad inicial con la que se etiqueta el tweet, por ejemplo 'P+':

```
P+
P
NEU
N
N+
NONE
Write the Polarity: P+
```

Una vez creados los tweets, estos se escriben como una línea (string), cada uno en un fichero distinto, que lleva por nombre 'tweetn.txt', donde n es el nº del tweet, o séase, el orden en que se ha escrito. El formato de la línea es el mismo que tienen los tweets en los ficheros de train y test, es decir, cada fichero contiene una línea que representa las principales características del tweet separadas por el carácter ';'.

En seguida veremos cómo adaptar al proceso a este modo de funcionamiento, ya que inicialmente estaba pensado para procesado de tweets en grandes grupos.

Opciones para seleccionar el nombre de los ficheros (opciones --conf_matrix_filename, --root_pathname, --root_filename --tfidfresults_filename:

Las principales son las siguientes:

- root_filename: Define el nombre del fichero que se va a utilizar para leer los tweets de test.
- root_pathname: Define la ruta donde se encuentra el fichero con los tweets de test.

Ejecución del BASELINE del proyecto (opción --baseline):

Esta opción es la que ejecuta el hilo principal del proyecto, que consiste en el preprocesado de los tweets y posterior clasificación de los mismos. Utilizando este enfoque, la característica mediante la cual el clasificador es capaz de distinguir polaridades es la medida del TF-IDF de los tweets.

Depuración del código (opción --debug):

Mediante esta opción, al ejecutarse el hilo principal se mostrará por pantalla información adicional de los distintos procesos que ocurren

en él, con el fin de que se pueda depurar el código a partir de la información obtenida.

Técnicamente, al utilizar esta opción lo que ocurre es que se ejecutan las instrucciones contenidas en todos los condicionales 'if debug':

```
if debug:
    print '[' + str(k) + '][' + word + '][' + '%.6f' % auxWordTFIDF + ']'
    #getchar()
```

Esta parte del código hace que durante la ejecución del baseline, mientras se ejecuta el proceso de extracción del TFIDF se muestre por pantalla el término procesado junto con su medida, lo cuál nos resultará útil a la hora de comprobar si la medida se está tomando correctamente del diccionario. Por ejemplo, en la ejecución del proceso para el tweet 'hola me parece estupendo' (introducido de forma manual), cuando se mapean los términos del tweet con el diccionario de TFIDF correspondiente a la clase P+ se mostrará por pantalla:

```
[0][hola][0.000107]
[3][estupendo][0.000291]
```

Que coinciden con los valores que tienen esos términos en el diccionario, por lo que hemos comprobado que se están extrayendo correctamente.

Ejecución Rápida (opción --fast_debug):

Permite una ejecución rápida del proceso de clasificación de tweets, con el fin de obtener resultados casi inmediatos, que no son del todo representativos del funcionamiento del clasificador pero pueden ser útiles para encontrar posibles errores en la ejecución del proceso.

Para ello, el entrenamiento del clasificador se realiza con 6000 tweets, y las pruebas se reducen únicamente a 1000 tweets (de aproximadamente 66000), con lo cual el procesado es mucho más rápido.

Para los tweets de test, el proceso SplitAllTweetsIntoWords tarda 6.69 segundos:

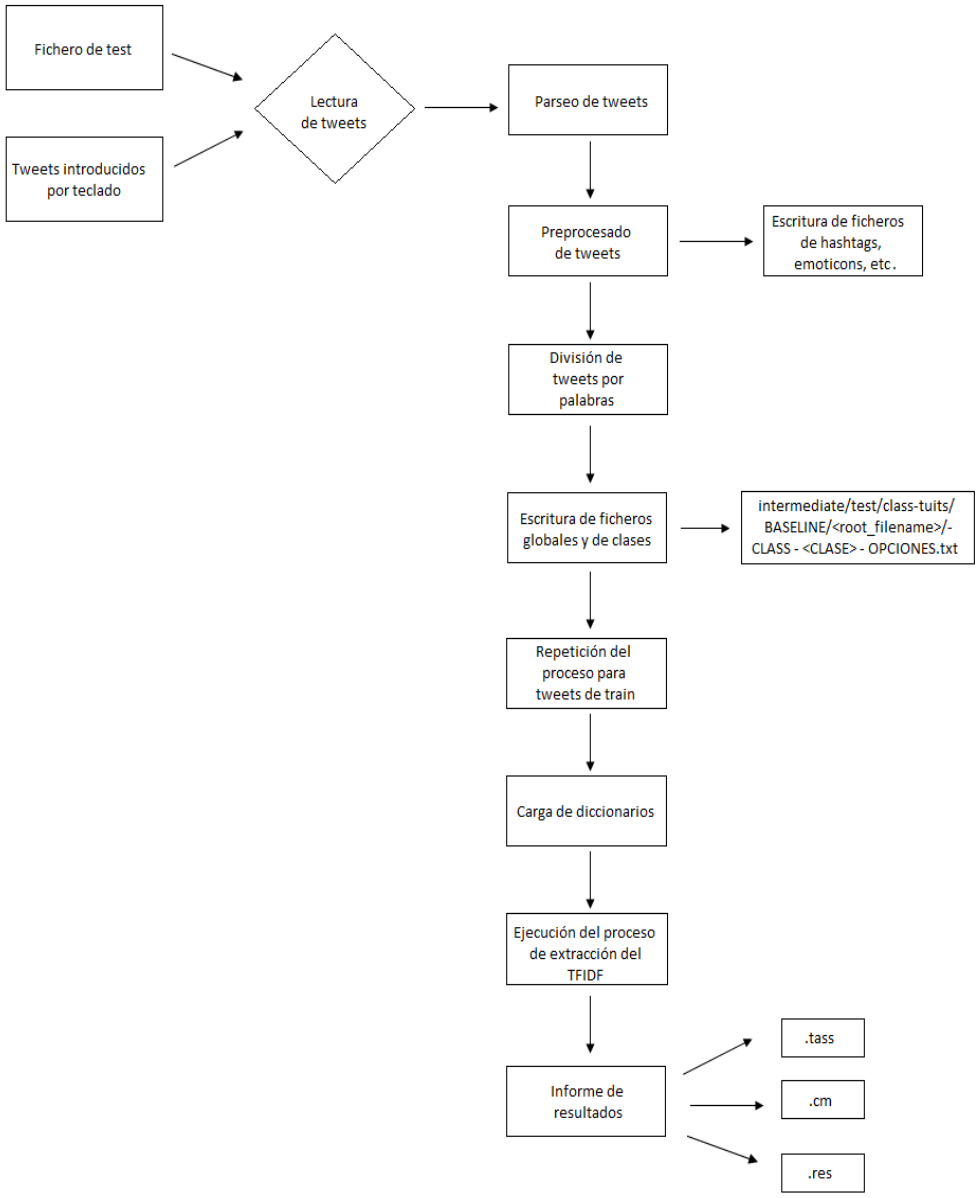
```
[PROCESS SplitAllTwitsIntoWords COMPLETE FOR 1000 TWITS][ELAPSED TIME 6.29 s (0.
10 min)]
[PROCESS ProcessPossibleQuotes COMPLETE FOR 1000 TWITS][ELAPSED TIME 0.04 s (0.0
0 min)]
```

Mientras que sin la ejecución rápida, para finalizar este mismo proceso se tardan aproximadamente 7 minutos:

```
[PROCESS SplitAllTwitsIntoWords FOR 60798 TWITS STARTS NOW]

[60000 TWITS COMPLETED OUT OF 60798][LAST 1000 IN 6.331333 s][ELAPSED TIME 378.6
725 s][END ESTIMATED IN 0.00 s (0.00 min)]
```

Una vez seleccionadas las opciones, comienza la ejecución del hilo principal, que sigue el siguiente proceso:



A continuación profundizaré en los pasos que realiza el hilo principal.

3.2 Flujo de trabajo del hilo principal (TASS2015-weka-testing.py)

PASO 1: LECTURA DE TWEETS DE PRUEBA:

Existen 2 opciones:

- Leer todos los tweets desde el mismo fichero.

- Opción 'single_tweet': Permite al usuario introducir tweets por teclado. Cada tweet introducido se escribe como una línea (string) en un fichero distinto (tweetn.txt) donde n es el nº del tweet.

PASO 2: PARSEO A ESTRUCTURA 'TWEET':

Es decir, se extraen los datos de dichas líneas y se integran en una estructura 'TWEET' mediante distintos métodos (principalmente el método 'GetFeatures').

Una vez se han parseado todos los tweets, se guardan en el array 'ParsedTweets'. Este array contendrá en cada posición una estructura tweet que a su vez lleva asociada mediante sus atributos las características del tweet en cuestión. Pues bien, partiendo de esta base el paso siguiente consiste en el preprocesado del texto del tweet:

PASO 3: PREPROCESADO DEL TEXTO:

Conceptualmente, el preprocesado de texto pasa por detectar distintos caracteres y fórmulas características del uso de twitter: hashtags, RT's, citas, emoticonos, urls... además en esta parte del código se incluyen otras utilidades como por ejemplo la detección de ironías.

El preprocesado se realiza así:

1. Se divide el texto del tweet por palabras (mediante el método splitTweetWords)

2. Se ejecutan los métodos de detección de hashtags, ironías, emoticonos, etc.

3. Una vez realizado el preprocesado, se sustituye en el tweet el texto susceptible de ser una de las fórmulas comunes por valores concretos, como se ha comentado anteriormente. Por ejemplo para el tweet "Hola me parece estupendo":

```
Twit labels:
[hashtagffm]
[TWIT][1][UPDATED!!!]
[NEW][Hola me parece hashtagffm]
```

4. Por último, dependiendo de las opciones se escriben los tweets con el texto ya preprocesado a ficheros .txt :

   - Hay varios ficheros, dependiendo de las opciones seleccionadas, que guardarán patrones detectados tales como emoticonos, hashtags, urls, ironías, etc.
     Se encuentran en la ruta:
     "REPO_TASS/intermediate/test/preprocess/ BASELINE/PARSED" y siguen este formato en cuanto a su nombre:

     <root_filename> - <patrón> - BASELINE – <DIC_SIZE> - OPCIONES.txt

   - Hay una serie de ficheros (tantos como clases haya) que contienen cada uno los tweets asociados a una clase a partir de la polaridad, a partir de los cuales se puede calcular el TF.
     Se encuentran en la ruta:
     "REPO_TASS/intermediate/test/class-tuits/BASELINE" y siguen este formato en cuanto a nombre:

     <root_filename>/ - CLASS - <CLASE> - OPCIONES.txt

PASO 4: REPETICIÓN DEL PROCESO PARA TWEETS DE TRAIN

PASO 5: CARGA DE DICCIONARIOS:

Se cargan todos los diccionarios disponibles después del procesado de los tweets.

A partir de aquí comienza el verdadero proceso de clasificación de los tweets en función de su polaridad, que hará una estimación de la polaridad de los mismos, que posteriormente será comparada con la etiqueta que tenían inicialmente. En caso de que coincidan se considerará acierto, en caso de que no lo hagan será un error. Se hará un porcentaje de errores y aciertos para tener una estadística

indicativa de cómo de bueno es nuestro clasificador de polaridad de tweets.

En el siguiente apartado, se explicará en profundidad cuál es el clasificador de polaridad propuesto. Como adelanto, comentar que está basado en las medidas TF-IDF de los términos de los tweets.

3.3 Flujo de trabajo del proceso de clasificación:

Determinar la polaridad de un tweet supone desde el punto de vista técnico la clasificación del mismo, pudiendo pertenecer el tweet a 6 clases:

P+ -> Muy positivo

P-> Positivo

NEU -> Polaridad neutra

NONE -> Sin polaridad

N -> Negativo

N+ -> Muy negativo

El algoritmo utilizado para la clasificación de los tweets incluye el cálculo de varios indicadores:

- El TF (Term Frequency), es la frecuencia de un término dentro de un documento.
  Para el caso de análisis del sentimiento, un documento equivale a un conjunto de tweets que pertenecen a una determinada clase (P.ej: el TF del término 'hola' para la clase 'P+' sería el nº de veces que aparece el término dentro de un conjunto de tweets etiquetados como P+). Normalmente, el cálculo del TF de un término se realiza de la siguiente manera:

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d) : w \in d\}}$$

  Siendo 'f(t,d)' la frecuencia del término en una clase y 'f(w,d)' el término que más se repite dentro de la misma. Dividir entre la frecuencia del término que más se repite tiene sentido, puesto que de esta manera el cálculo queda normalizado respecto a la longitud del documento, de manera que la medida

del tf no se verá afectada por el nº de palabras que contenga cada conjunto.

Entonces, podríamos intentar determinar la polaridad de un tweet a partir de los TF de las palabras que lo conforman. Una forma de proceder podría ser la siguiente:

- En primer lugar, calcularíamos el TF de cada término para todas las clases disponibles (P. ej: el TF del término 'hola' para las clases 'P', 'NONE' y 'N', suponiendo que sólo se dispone de esas tres clases).
- Sumaríamos para cada una de las clases los TF obtenidos para todos los términos del tweet (P.ej: sumar los TF de los términos del tweet para la clase 'P', después para la clase 'NONE' y por último para la clase 'N').
- Por último, el tweet se etiquetaría con la clase para la que la suma resultase máxima.

Sin embargo, utilizar un algoritmo que únicamente extraiga el TF total de cada tweet es un error, ya que esta medida tiene ciertas limitaciones:

- Cuando utilizamos como medida de polaridad el TF, estamos viendo los tweets con un enfoque de 'bag of words', es decir, cada tweet es un conjunto de términos en el que el orden de los mismos no tiene importancia, al contrario que el nº de ocurrencias de cada término (P.ej: la frase "lo mismo me da" es idéntica desde este punto de vista a la frase "me da lo mismo"). Por tanto es de suponer que varios tweets con un 'bag of words' similar muy posiblemente obtengan los mismos resultados respecto a su polaridad. Esto no siempre es conveniente, ya que un mismo conjunto de palabras puede tener distintas connotaciones dependiendo del contexto y el tono en que se utilicen (P.ej: una frase puede estar constituida por casi las mismas palabras cuando se pretende expresar un tono serio y cuando tiene un sentido irónico).

- Además, existen palabras que aparecerán con mucha frecuencia en los tweets, y que sin embargo no tienen mucha relevancia en cuanto a expresión de sentimientos, por lo que no debería tenerse tanto en cuenta su TF. Son las conocidas como 'stop words' (puede consultarse una lista de

'stop words' en español en el siguiente enlace: http://www.ranks.nl/stopwords/spanish), y conviene descartar su medida. De modo que, a la hora de calcular la polaridad de los términos, hay que tener en cuenta que no basta con que estos se repitan con mucha frecuencia dentro de una clase, sino que además deben repetirse poco dentro de la lista completa de tweets, o lo que es lo mismo, se deben penalizar de alguna manera aquellas medidas de TF de

términos que no sean específicos de una cierta clase. Es por esto por lo que, además del TF se calcula otro indicador denominado IDF, que se define a continuación.

- El IDF (Inverse Term Frequency), es la frecuencia de un término dentro de un conjunto de documentos. Para nuestro caso sería el equivalente al nº de veces que aparece un término dentro del conjunto de todos los tweets, sin hacer distinciones de polaridad. Es decir, se calcula como:

$$idf\ (t, D) = \ log\ \frac{|D|}{|\{d\ \in D : t\ \in D\}|}$$

Siendo 't' el término, 'D' el nº total de tweets, y 'd' el nº total de tweets que contienen el término en cuestión. Nótese que el IDF será menor cuanto más frecuente sea el término en la colección de documentos.

En conclusión, mezclando estos indicadores obtenemos el TF-IDF, que es una de las características que se han tenido en cuenta en la estrategia de clasificación propuesta y que se calcula como el producto de ambos:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

De manera que aquellos términos con mayor TFIDF para una clase serán muy frecuentes en los tweets etiquetados como pertenecientes a dicha clase y tendrán baja frecuencia en los tweets de las clases restantes.

3.4 Descripción del proceso "TEST-TFIDF_MEASURES"

Este es el proceso que obtiene las medidas TFIDF de los tweets a partir de los diccionarios de polaridad disponibles.

A continuación se muestra un ejemplo para el siguiente tweet: 'Hola me parece estupendo', que ha sido introducido por teclado.

1. En primer lugar, el proceso recorre el tweet por términos e intenta encontrar cada uno en todos los diccionarios de TFIDF (es decir, busca si el término existe en el diccionario de términos 'N+', después en el de términos 'N'… y así sucesivamente hasta llegar a 'P+') .

2. Dependiendo de la existencia o no del término en el diccionario en el que se está buscando hay 2 opciones:
   - Si el término existe en el diccionario se incrementa el nº de términos útiles en 1.
   - Si el término no existe se le asigna el valor 'unk'

   Mientras se ejecuta el programa se puede observar cuántos términos han sido encontrados correctamente para la clase y cuáles son:

```
[Test_TFIDF_Measures][Mapping twit 1 for class P+]
[0][hola][->][hola]
[1][me][->][unk][KeyError!!!]
[2][parece][->][unk][KeyError!!!]
[3][estupendo][->][estupendo]
[NEW][ holaunkunk estupendo][2 OK out of 4]
```

3. A los términos que han sido encontrados correctamente en el diccionario de una de las clases (términos útiles), se les asigna como valor de TF-IDF el que tengan en el mismo. Estos valores son mostrados por pantalla:

```
[0][hola][0.000107]
[3][estupendo][0.000291]
```

4. Para cada clase, se suman los TF-IDF de los términos útiles y se comparan los totales obtenidos. El tweet será etiquetado como perteneciente a la clase para la que se obtenga la máxima suma total. Para el ejemplo:

```
[Twit][1][Hola me parece estupendo]
[GROUNDTRUTH][P]
[('P+', 0.00039870268403960764), ('NONE', 0.0001350574945122404),
('P', 0.00012505349278549326), ('N+', 0), ('N', 0), ('NEU', 0)]
aux_gtruth_class = [P]
aux_winner_class = [P+][0.000399]
```

La clase ganadora es 'P+', ya que es la que tiene un TF-IDF máximo en comparación al resto. Sin embargo, se puede observar que el tweet estaba etiquetado como 'P' (aux_gtruth_class = [P]), por lo que la clasificación sería errónea. Esto se debe a que el tamaño de los diccionarios seleccionado para el experimento anterior, así como el n de tweets de entrenamiento no era el más adecuado.

Una vez se han obtenido los resultados del cálculo del TF-IDF, se genera el informe de resultados, que consta de tres ficheros con distinto formato:

1. Fichero .cm(intermediate/test/results/BASELINE/ <conf_matrix_filename> - OPCIONES.txt)

   Contiene la matriz de confusión resultante del experimento. Esta matriz representará mediante sus filas la clase real de los tweets, y mediante sus columnas el nº de tweets que hay de cada clase real, repartidos en:

   - Etiquetados correctamente: Aquellos en los que la clase de la fila y la clase de la columna coinciden. O, dicho de otro modo, aquellos que se encuentren en la diagonal principal de la matriz de confusión.
   - Etiquetados erróneamente: Aquellos en los que la clase de la fila y la clase de la columna no coinciden.

A continuación se muestra la matriz obtenida para un experimento rápido (utilizando únicamente 100 tweets para probar):

```
[CONFUSION MATRIX]
        N+      N       NONE    NEU     P       P+
N+      0       2       4       0       0       0
N       1       0       9       0       0       0
NONE    2       1       56      0       0       1
NEU     0       0       0       0       0       0
P       0       0       2       0       0       0
P+      0       0       18      0       0       4
[ACCURACY][60.00 (50.40, 69.60)][60 OK / 40 WRONG]
```

Un ejemplo de cómo interpretar esta matriz es el siguiente: Dentro del conjunto de tweets de 'test', se tenían 60 pertenecientes a la clase 'NONE', de los cuales 56 se han etiquetado correctamente y 4 no (hay un par de ellos etiquetados como 'N+', uno como 'N' y otro como 'P+'.

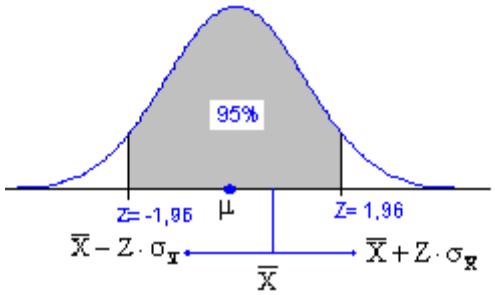
Además de la matriz, el fichero incluye una serie de puntuaciones, como se muestra en la imagen anterior:

- Porcentaje medio de éxito de la estimación de polaridad. Se calcula como el nº de tweets etiquetados correctamente dividido por el nº total de tweets del experimento. En el ejemplo es de 0.6 (60%).

- Intervalo de confianza de la estimación. Se trata de un rango de valores que puede tomar el porcentaje de éxito. Como se ha comentado anteriormente, el porcentaje de éxito es un valor promedio obtenido tras ajustar el clasificador mediante una validación cruzada 10-fold. Es decir, en cada una de las 10 iteraciones se puede dar un valor distinto de porcentaje de éxito, y estos valores estarán dentro de un intervalo de confianza, que se calcula como:

$$1.96 * \sqrt{\frac{Nº\ tuits\ correctos * Nº\ tuits\ erróneos}{Nº\ de\ tuits\ total}}$$

NOTA: Se multiplica por 1.96 para obtener un nivel de confianza del 95 %. Si vemos el porcentaje de éxito como una gaussiana, |z|=1.96 es el valor de z que deja entre el intervalo al 95% de los valores medios del porcentaje de éxito.



Es decir, se multiplica por 1.96 para que el 95% de las veces que se realice el experimento, los porcentajes de éxito queden en el intervalo de confianza establecido. En el ejemplo el 95% de veces que se ejecute el experimento con datos que sigan la misma proporción en cuanto a polaridad, el porcentaje de éxito estará comprendido entre 50.4 y 69.60, que es el intervalo de confianza.

- Nº de tweets etiquetados correctamente / Nº de tweets etiquetados erróneamente.

2. Fichero.res (intermediate/test/results/BASELINE/ <tfidf_results_filename> - OPCIONES.txt)

   Contiene información acerca del nº de términos útiles que se han encontrado en el tweet para cada clase, y cuál es el TFIDF de los mismos en dicha clase.

Por ejemplo, tras finalizar el proceso para el tweet 'Hola me parece estupendo' (introducido por teclado) el fichero de resultados del tweet contiene lo siguiente:

TWIT	N+	N+_UW	N
1	0.00000000	0	0.00000000
N_UW	NONE	NONE_UW	NEU
0	0.00013506	1	0.00000000
NEU_UW	P	P_UW	P+
0	0.00012505	1	0.00039870
P+_UW	GROUNDTRUTH	RESULT	TWIT_WORDS
2	P	P+	4

La interpretación de esta tabla es la siguiente:

- El campo 'TWEET' es un identificador del tweet procesado. Tiene sentido cuando se genera un único informe que contiene los resultados de todos los tweets de 'test'.

- Los campos que acaban en '_UW' representan el nº de términos útiles encontrados para una cierta clase. Por ejemplo: P+_UW =2. Esto significa que se han encontrado 2 palabras en el tweet que están en el diccionario de TFIDF correspondiente de la clase 'P+' (las palabras 'hola' y 'estupendo'.

- Los campos con los nombres de las clases tienen como valor la suma de los TF-IDF de los términos útiles encontrados en cada una. Por ejemplo, para la clase 'P+' en este tweet el TF-IDF vale '0.00039870'. Este valor es la suma de los TF-IDF de los términos útiles:

```
[0][hola][0.000107]
[3][estupendo][0.000291]
```

  NOTA: Como el valor anterior es máximo respecto al resto de sumas, la etiqueta que se asignará al tweet será 'P+'.

- El campo 'GROUNDTRUTH' contiene la etiqueta inicial asignada al tweet.
- El campo 'TWEET_WORDS' contiene el nº total de palabras que tiene el tweet.

- Por último, el campo 'RESULT' contiene la etiqueta asignada al tweet una vez finalizado el proceso.

3. Fichero.tass: Fichero con 3 columnas, que contienen la siguiente información:

   1ª columna: Identificador del tweet
   2ª columna: Etiqueta asignada tras el proceso
   3ª columna: Nivel de confianza de la estimación

   Por ejemplo, para una ejecución del proceso con 1000 tweets de prueba y 6000 de entrenamiento, el fichero .tass incluye resultados del tipo:

```
142379080808013825      P+      1
142396855916044288      N       1
142415001527918592      NONE    1
142482752330153984      N+      1
```

4. Resultados obtenidos y Conclusiones

RUN 1: Se ha ejecutado el proceso sin realizar el procesamiento tweet a tweet, y con las opciones activas: NUMPAR, REPCHARS, EMOTICONS, FAST_DEBUG para un tamaño de diccionario de 500 términos.

Se han obtenido los siguientes resultados para 1000 tweets de test, entrenando con 6000 tweets de train utilizando 6 clases:

```
[CONFUSION MATRIX]
        N+      N       NONE    NEU     P       P+
N+      32      9       9       0       1       3
N       20      51      45      12      11      6
NONE    32      38      238     16      41      36
NEU     3       6       4       0       2       1
P       2       0       3       0       12      3
P+      11      12      122     8       49      162
[ACCURACY][49.50 (46.40, 52.60)][495 OK / 505 WRONG]
[STATISTICALLY SIGNIFICANT IF COMPARED WITH zeroR!!!! :-)))))]

[N+]
[RECALL][59.26][32 OK / 22 WRONG][54 TOTAL]
[PRECISION][32.00][32 TRUE / 68 FALSE]

[N]
[RECALL][35.17][51 OK / 94 WRONG][145 TOTAL]
[PRECISION][43.97][51 TRUE / 65 FALSE]

[NONE]
[RECALL][59.35][238 OK / 163 WRONG][401 TOTAL]
[PRECISION][56.53][238 TRUE / 183 FALSE]

[NEU]
[RECALL][0.00][0 OK / 16 WRONG][16 TOTAL]
[PRECISION][0.00][0 TRUE / 36 FALSE]

[P]
[RECALL][60.00][12 OK / 8 WRONG][20 TOTAL]
[PRECISION][10.34][12 TRUE / 104 FALSE]

[P+]
[RECALL][44.51][162 OK / 202 WRONG][364 TOTAL]
[PRECISION][76.78][162 TRUE / 49 FALSE]
```

Y para 4 clases:

```
[CONFUSION MATRIX]
        N       NONE    P
N       112     66      21
NONE    79      258     80
P       25      133     226
[ACCURACY][59.60 (56.56, 62.64)][596 OK / 404 WRONG]
[STATISTICALLY SIGNIFICANT IF COMPARED WITH zeroR!!!! :-)))))]

[N]
[RECALL][56.28][112 OK / 87 WRONG][199 TOTAL]
[PRECISION][51.85][112 TRUE / 104 FALSE]

[NONE]
[RECALL][61.87][258 OK / 159 WRONG][417 TOTAL]
[PRECISION][56.46][258 TRUE / 199 FALSE]

[P]
[RECALL][58.85][226 OK / 158 WRONG][384 TOTAL]
[PRECISION][69.11][226 TRUE / 101 FALSE]
```

Estos resultados son negativos para la clasificación de 6 etiquetas, debido al pequeño tamaño de los diccionarios**. Sin embargo, los resultados obtenidos para 4 clases son de casi un 60%, que es un porcentaje aceptable.** En este caso, aunque los diccionarios sean más pequeños, el nº de clases también lo es, por lo que los tweets 'P+' se etiquetarán como 'P' y los tweets 'N+' se etiquetarán como 'N', aumentando el porcentaje de acierto.

RUN 2: Se ha ejecutado el proceso igual que en la ejecución anterior pero aumentando el tamaño de los diccionarios a 4700, siendo el resultado peor al producirse sobreajuste.

RUN 3: Se ha ejecutado el proceso sin FAST-DEBUG, siendo terminado por el sistema al no soportar la carga computacional del mismo.

Conclusions:

- I must improve on my analysis, because I had good results for one of the tasks but not for both. The available dictionaries are not enough and neither the processing costs of the algorithm.

- I have to explore other alternatives, for example, trying to use more indicators, and investigate how can I use the single tweet option to send the tweets to a cluster.

I'll try to figure all of it as soon as possible to improve the results.

5. Summary:

5.1 Introduction:

Nowadays, Internet has evolved into "Web 2.0", due to the boom in services that keep information on the network. So we can extract a lot of information from the users. Applicating Sentiment Analysis to this information can be very useful to several areas, so this research field is rising next to the Web 2.0 services.

The Sentiment Analysis consists in trying to determine the characteristics of a target and extract the polarity of each one. This means classifying opinions through opinion mining techniques. Like we said, this can be useful for example to know the strengths and areas for improvement points of a product.

In addition to a marketing-oriented use, we can use sentiment analysis to get social information on other topics of interest like policy, economy, hobbies, etc.

Within social networks, Twitter has become the main when it comes to review any topic, so researchers in the opinion mining and sentiment analysis field are devoting great efforts in analyzing tweets. As an example of this, in recent years the number of tasks and contests related to the analysis of feeling, like TASS Workshops, has grown considerably.

5.2 TASS Workshops:

Tass stands for Sentiment Analysis and opinion tracing in social networks (in Spanish). In these workshops a number of related to sentiment analysis tasks are proposed to encourage the design of classification algorithms, then comparing the different implementations of the systems developed by the participants.

This has a scientific approach, because it ultimately consists in getting, for each edition, the best social feeling analysis through the expressed opinions on Twitter. The proposed tasks are the following:

1. **Tweets classification depending on the topic they talk about**: It consists in correctly assign one or more topics to tweets. The topics are the following:

- Movies
- Sports
- Economy
- Entertainment
- Football
- Literature
- Music
- Others
- Politics
- Technology

2. **Aspects Detection:** It is about detecting all references to an entity in the tweets. For example:

   Jordan, @Air, AirJordan, MJ23, #JordanForever, Michael Jordan, Michel Jordan, @Jordan23, @MJBULLS, michael jordan, @Jordan_The_Best, @Forever_23, etc.

3. **Aspects polarity Analysis:** It depends on Task 2, because it consists in adding polarity to the detected aspects on that task. For example, Jordan will have positive and negative references, and we must decide (using TFIDF for example), which class is the winner, ie, what opinion type is more frequent when people talk about Jordan.

4. **Tweets Sentiment Analysis:** It consists in determining tweets polarity using different techniques. For this task, there are 2 subtasks:

   1º) Distinguishing between 6 classes (N, N+, NEU, NONE, P, P+).

   2º) Distinguishing between 4 classes (N, NEU, NONE, P).

The proposed solution focuses on resolving this task, but I would like to continue developing the code in order to facing the other tasks.

5.3 Proposed Solution:

There are several useful indicators when it comes to extracting the polarity of the tweets:

- TF-IDF
- Lemmas
- POS (part of speech) tags
- Emoticons
- Discourse Level opinion mining
- Followers graph

Besides, there are many strategies. Experience has shown that in this task, the researchers have obtained the best results by training the clasifficator through machine learning, and combining this with the use of polarity lexicons.

I am going to use a classification scheme based on the words, lemmas and pre-processed n-grams TFIDF measures, starting with the creation of dictionaries from TFIDF frequencies within the terms in selectable size lexicons. I will take also into account other indicators such as emoticons.

1 on 1 processing is necessary in order to reduce the processing costs, and for the training the classifier is going to be trained using the 10-folds-cross-validation technique to adjust its parameters.

Finally, I am going to estimate a confidence interval for a 95% confidence level in the values of the estimation.

5.4 Proposed Algorithm Design:

The main thread executes the following processes:

1. Reading the test file/Introducing test tweets manually (--single_tweet option)

2. Parsing the test tweets

3. Preprocessing the test tweets and writing new files for saving detected patterns (like emoticons, hashtags, etc)

4. Repeating steps 1, 2, 3 for the train tweets.

5. Loading polarity lexicons (globals and TFIDF).

6. Calling to PROCESS_TFIDF_MEASURES, which executes the following sub-processes:

    6.1 Mapping tweets for each class.

    6.2 Obtaining TFIDF for the terms in each class, tagging each term with the label which has obtained the maximum TFIDF Measure for the term.

    6.3 Obtaining TFIDF for the tweets through the terms measures, by grouping terms TFIDF Measures values and taking the one class which has the maximum value.

    6.4 Reporting the results.

    6.5 Saving the results in 3 new files (.cm, .tass and .res).

5. 5 Obtained results and conclusion

Obtained results:

After 3 runs I have seen that the classificatory works decently for the 4 class sentiment analysis, because with only 500 terms in each dictionary I have obtained an average result of 60 %.

Nevertheless, the classificatory doesn't work correctly for the 6 class sentiment analysis, because the available dictionaries are not enough in this case, and the computational costs are too high.

So, I must investigate how to use the single_tweet mode to divide the processing cost of the test and training files by sending the processed tweets to a cluster, where the algorithm will run much faster.

Besides, I have to run the experiment using other features and more preprocessing in order to obtaining better results.

Conclusion:

To design a machine learning- related algorithm is always difficult, but very gratifying when it works. It has been very exciting for me to participate on this project for finishing my degree, and I am happy to keep working on this algorithm to improve it.

6. Referencias:

www.wikipedia.org
http://www.daedalus.es/TASS2014/papers/8.Elhuyar.pdf
http://www.daedalus.es/TASS2014/papers/6.ELiRF_UPV.pdf
http://das.sagepub.com/content/4/2/133.short
http://es.slideshare.net/dav009/label-propagation-semisupervised-learning-with-applications-to-nlp
https://people.cs.pitt.edu/~wiebe/pubs/papers/coling08.pdf
http://cran.r-project.org/web/packages/twitteR/twitteR.pdf